Introduction to Serverless PHP

Rob Allen

April 2019

Platform options

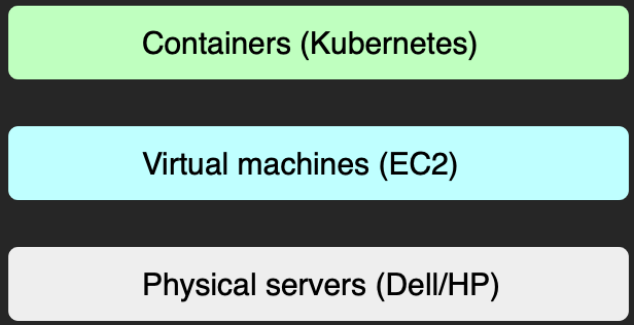
Physical servers (Dell/HP)

Platform options

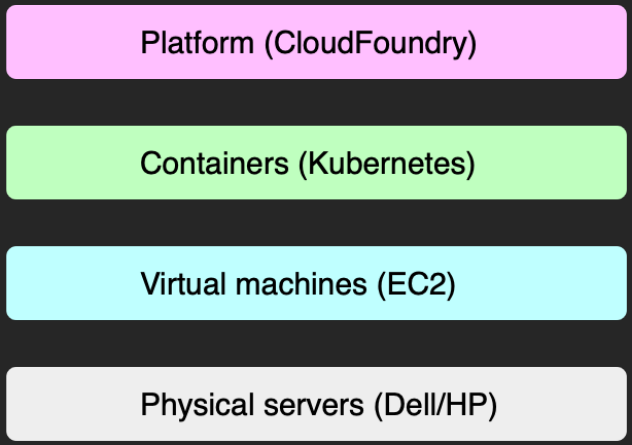
Virtual machines (EC2)

Physical servers (Dell/HP)

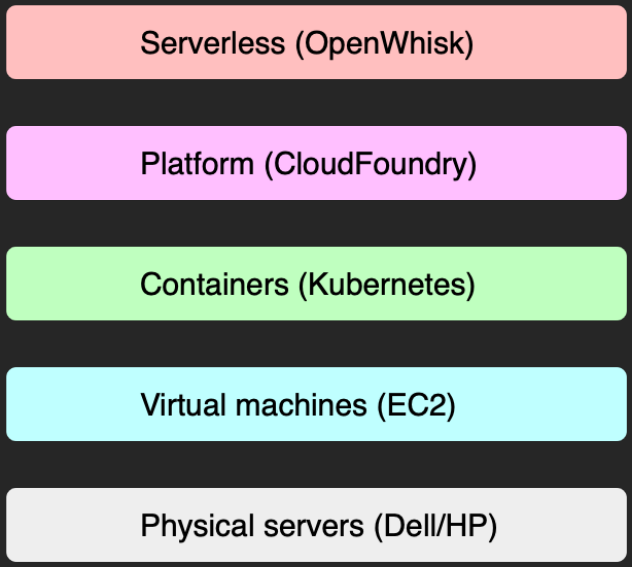
Platform options



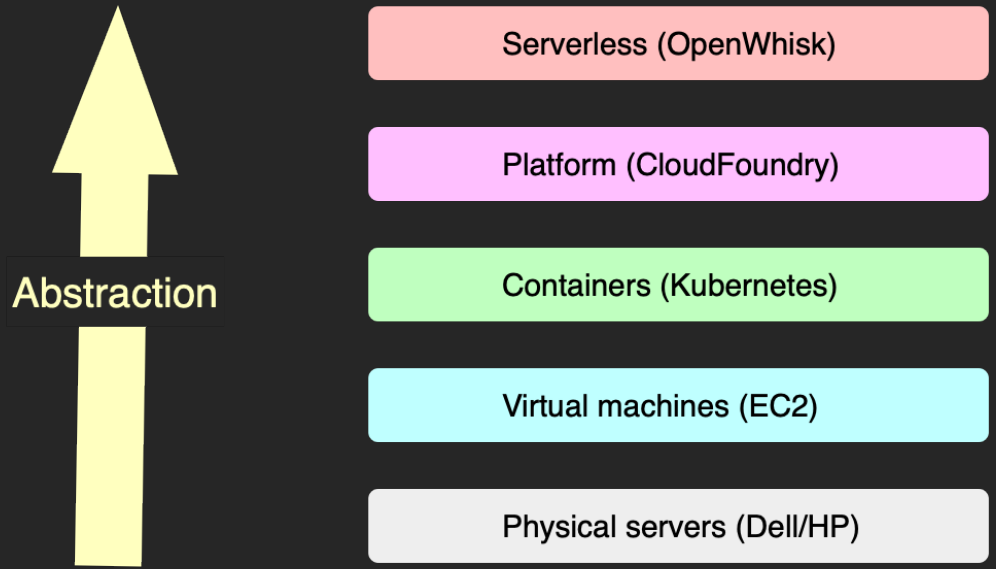
Platform options



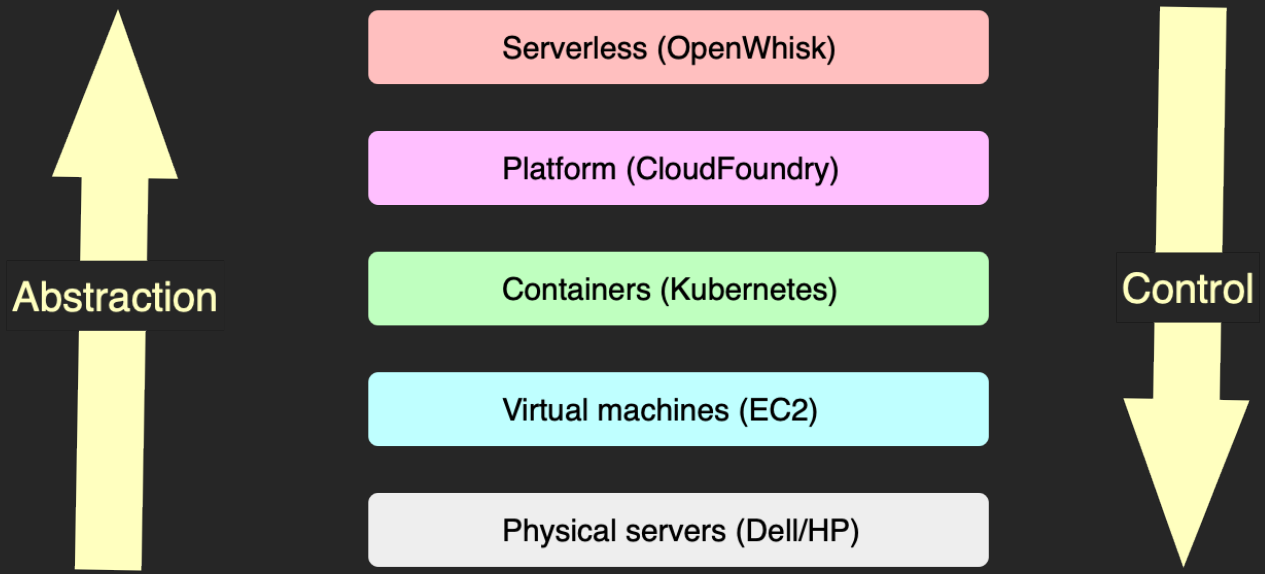
Platform options



Platform options



Platform options



Serverless

Serverless is all about composing software systems from a collection of cloud services.

With serverless, you can lean on off-the-shelf cloud services resources for your application architecture, focus on business logic and application needs.

Nate Taggart, CEO Stackery

FaaS

- Your code
- Deployed to the cloud
- Runs when needed
- Scaled automatically
- Pay only for execution

Where are the servers?


@akrabat

Not Your PROBLEM!

Use-cases

Use-cases

Synchronous

- Service is invoked and provides immediate response (HTTP requests: APIs, chat bots)

Use-cases

Synchronous

Service is invoked and provides immediate response
(HTTP requests: APIs, chat bots)

Asynchronous

Push a message which drives an action later
(web hooks, timed events, database changes)

Benefits

Benefits

- No need to maintain infrastructure

Benefits

- No need to maintain infrastructure
- Concentrate on application code

Benefits

- No need to maintain infrastructure
- Concentrate on application code
- Pay only for what you use, when you use it

Benefits

- No need to maintain infrastructure
- Concentrate on application code
- Pay only for what you use, when you use it
- Language agnostic

Challenges

Challenges

- Start up latency

Challenges

- Start up latency
- Time limit

Challenges

- Start up latency
- Time limit
- State is external

Challenges

- Start up latency
- Time limit
- State is external
- Different way of thinking

When should you use serverless?

When should you use serverless?

- Responding to web hooks

When should you use serverless?

- Responding to web hooks
- PWA/Static site contact form, et al.

When should you use serverless?

- Responding to web hooks
- PWA/Static site contact form, et al.
- Additional features without extending current platform

When should you use serverless?

- Responding to web hooks
- PWA/Static site contact form, et al.
- Additional features without extending current platform
- Variable traffic levels

When should you use serverless?

- Responding to web hooks
- PWA/Static site contact form, et al.
- Additional features without extending current platform
- Variable traffic levels
- When you want your costs to scale with traffic

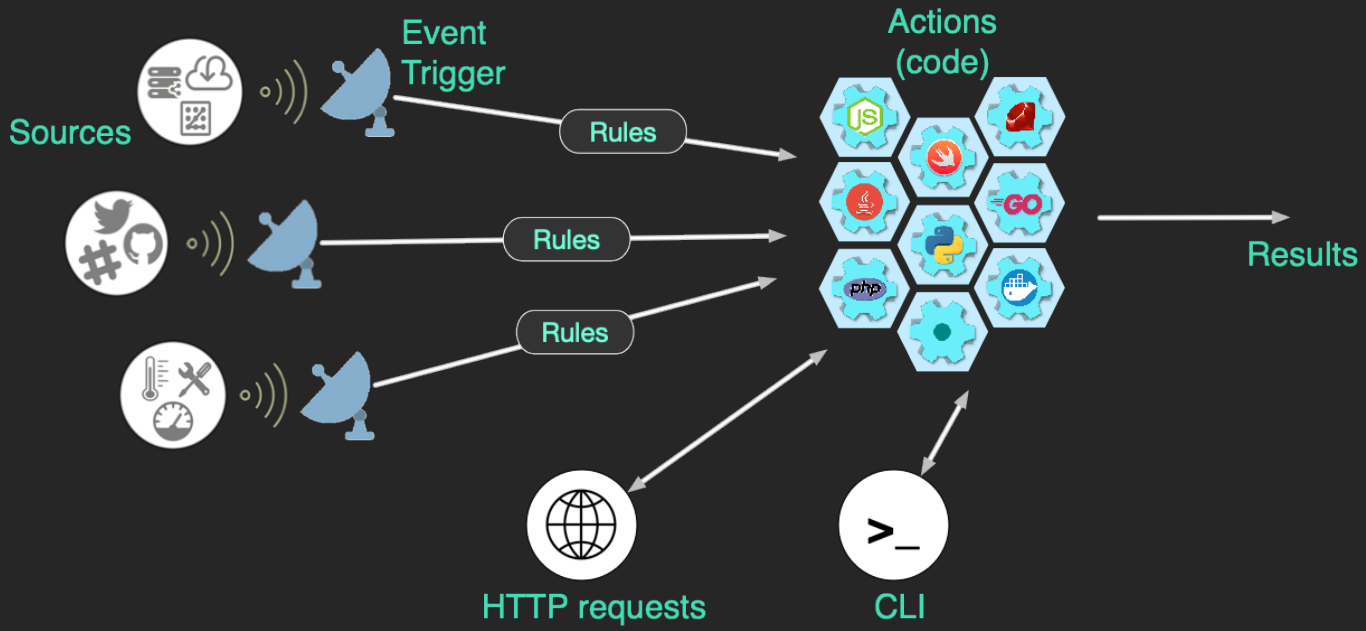
Serverless platforms

Serverless platforms with PHP support

APACHE

OpenWhisk

Concepts


Sources
Event
Trigger
Rules
Rules
Rules
Actions
(code)
Results
HTTP requests
CLI


JS

.NET
Core


php



GO

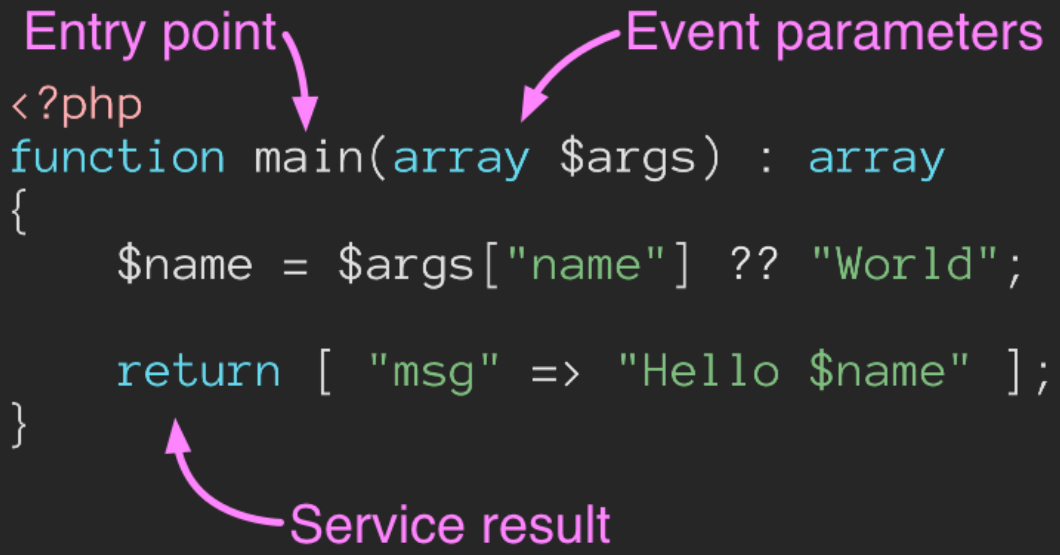
Java


JS
php
GO
Java

Hello world in PHP

```php
<?php
function main(array $args) : array
{
    $name = $args["name"] ?? "World";

    return [ "msg" => "Hello $name" ];
}
```

Hello world in PHP



Upload your action

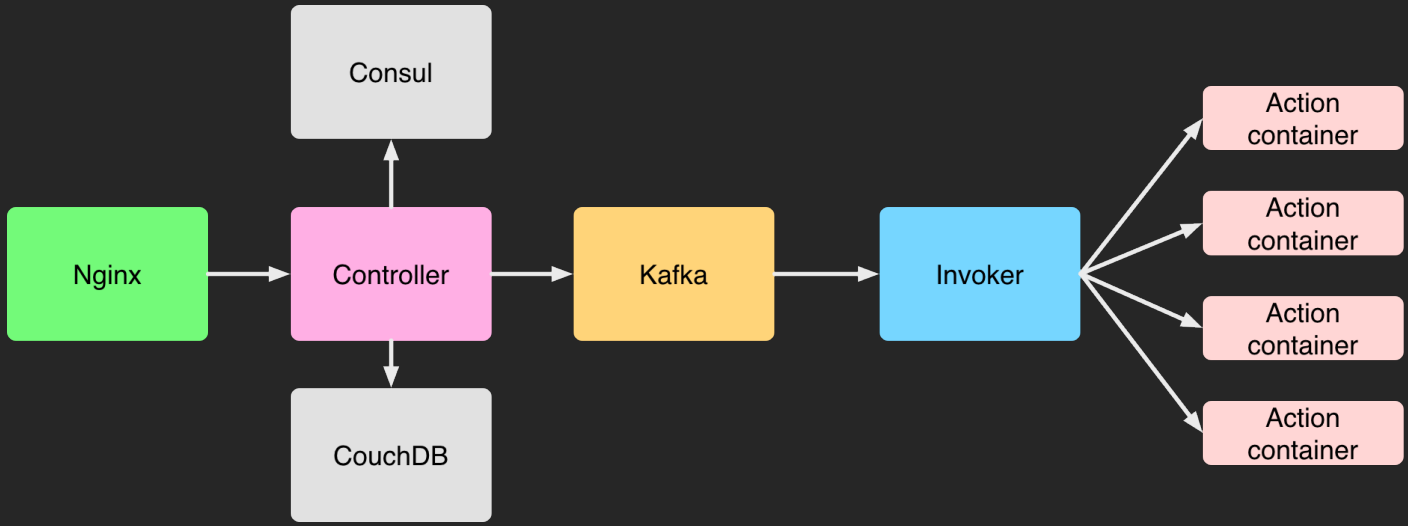
```
$ wsk action update hello hello.php
ok: updated action hello
```

Run your action

```
$ wsk action invoke hello --result
{
  "msg": "Hello World"
}
```

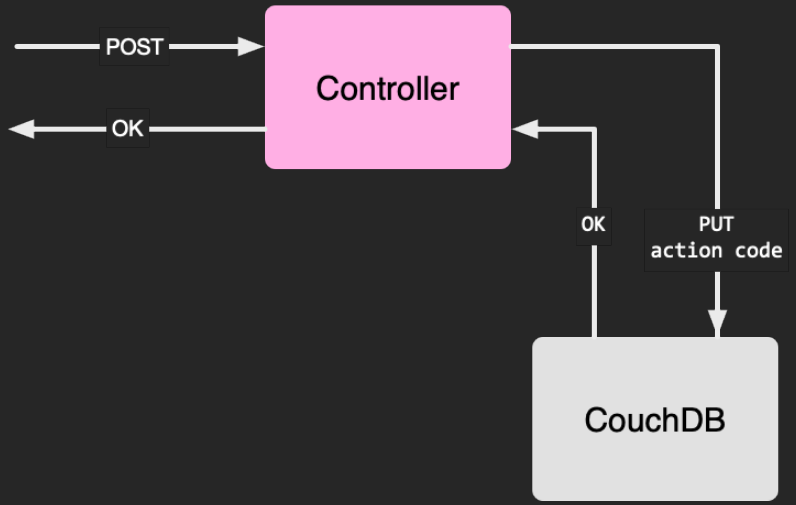
Under the hood

OpenWhisk's architecture



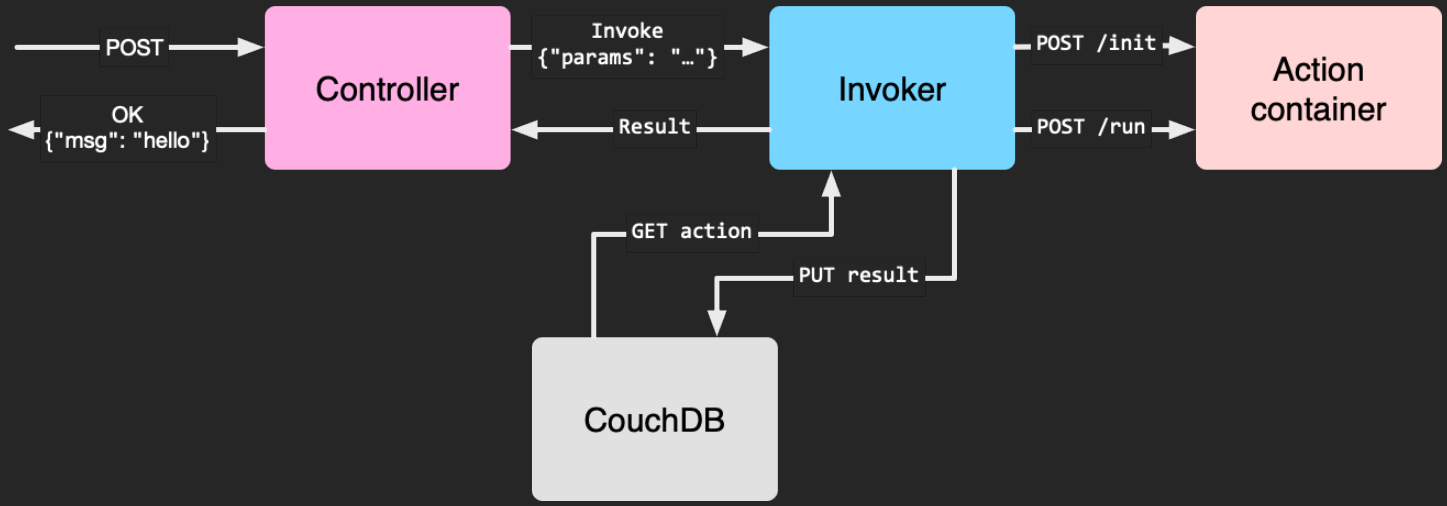
Create an action

```
$ wsk action create hello hello.php
```



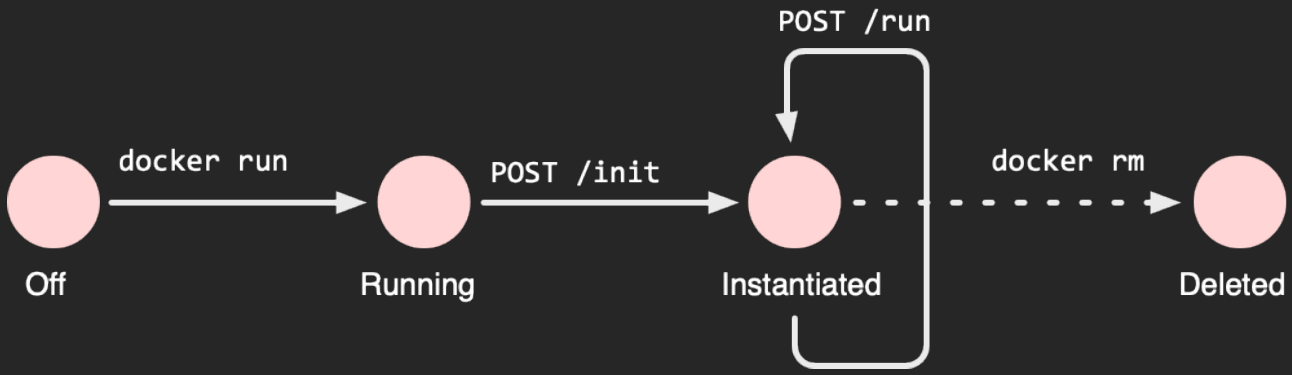
Invoke an action

```
$ wsk action invoke hello -r
```



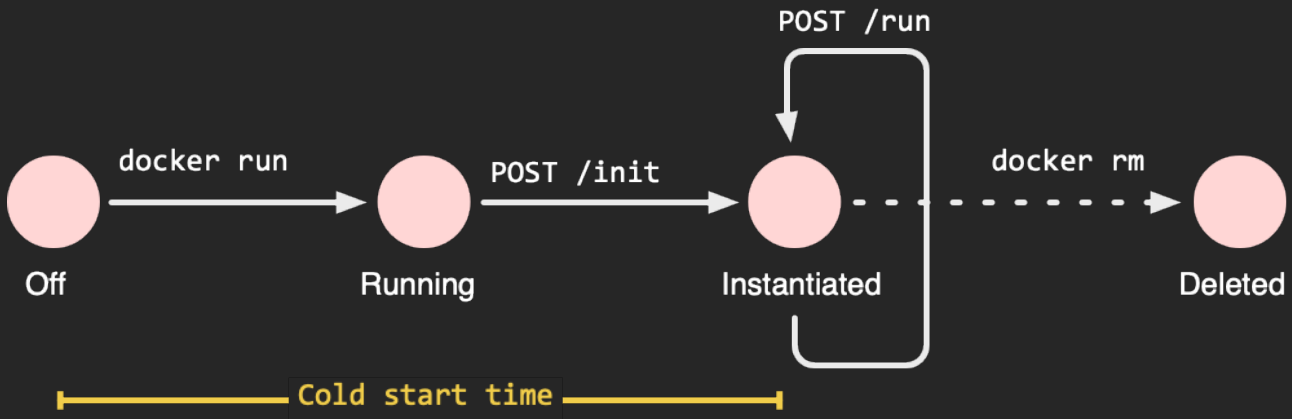
Action container lifecycle

- Hosts the user-written code
- Controlled via two end points: `/init` & `/run`



Action container lifecycle

- Hosts the user-written code
- Controlled via two end points: `/init` & `/run`




aws

AWS Lambda with PHP

Only *sensibly* possible since November 2018 with the introduction of *layers*

AWS Lambda with PHP

Only *sensibly* possible since November 2018 with the introduction of *layers*

Process:

1. Create a layer containing:
    1. the PHP executable
    2. Write a bootstrap script
2. Write the PHP function!

Full details: akrabat.com/lambdaphp

Bootstrap file

```
// layer/php/boostrap.php
while (true) {
  // get next event
  $eventPayload = $lambdaRuntime->getEventPayload();

  // execute handler
  $data = $handlerFunction($eventPayload);

  // send result
  $lambdaRuntime->sendResult($data);
}
```

Serverless Framework

Infrastructure as code

Manage using Serverless Framework

application manifest: `serverless.yml`

```yaml
service: hello-lambdaphp

provider:
  name: aws
  runtime: provided
  memorySize: 128

layers:
  php:
    path: layer/php
```

Manage using Serverless Framework

```
service: hello-lambdaphp

provider:
  name: aws
  runtime: provided
  memorySize: 128

layers:
  php:
    path: layer/php
```

Manage using Serverless Framework

```
service: hello-lambdaphp

provider:
  name: aws
  runtime: provided
  memorySize: 128

layers:
  php:
    path: layer/php
```

Hello World in Lambda

```php
function hello($eventData)
{
    $data = json_decode($eventData, true, 512,
        JSON_THROW_ON_ERROR);

    $name = $data['name']  ?? 'World';

    return json_encode(['msg' => "Hello $name"]);
}
```

Hello World in Lambda

```
function hello($eventData)
{
    $data = json_decode($eventData, true, 512,
        JSON_THROW_ON_ERROR);

    $name = $data['name']  ?? 'World';

    return json_encode(['msg' => "Hello $name"]);
}
```

Hello World in Lambda

```
function hello($eventData)
{
    $data = json_decode($eventData, true, 512,
        JSON_THROW_ON_ERROR);

    $name = $data['name']  ?? 'World';

    return json_encode(['msg' => "Hello $name"]);
}
```

Hello World in Lambda

```
function hello($eventData)
{
    $data = json_decode($eventData, true, 512,
        JSON_THROW_ON_ERROR);

    $name = $data['name']  ?? 'World';

    return json_encode(['msg' => "Hello $name"]);
}
```

Manage using Serverless Framework

```
functions:
  hello:
    handler: handler.hello
    layers:
      - {Ref: PhpLambdaLayer}
```

Deploy using Serverless Framework

```
$ sls deploy
Serverless: Packaging service...
Serverless: Uploading CloudFormation file to S3...

...

functions:
  hello: hello-lambdaphp-dev-hello
layers:
  php: arn:aws:lambda:eu-west-2:661969457706:layer:php:
Serverless: Removing old service artifacts from S3...
```

Invoke using Serverless Framework

```
$ sls invoke -f hello --data='{"name":"Rob"}'
{
    "msg": "Hello Rob"
}
```

Bref

Community maintained PHP runtime for AWS Lambda

```php
<?php

require __DIR__.'/vendor/autoload.php';

lambda(function (array $event) {
    $name = $event['name']  ?? 'World';
    return ['msg' => "Hello $name"];
});
```

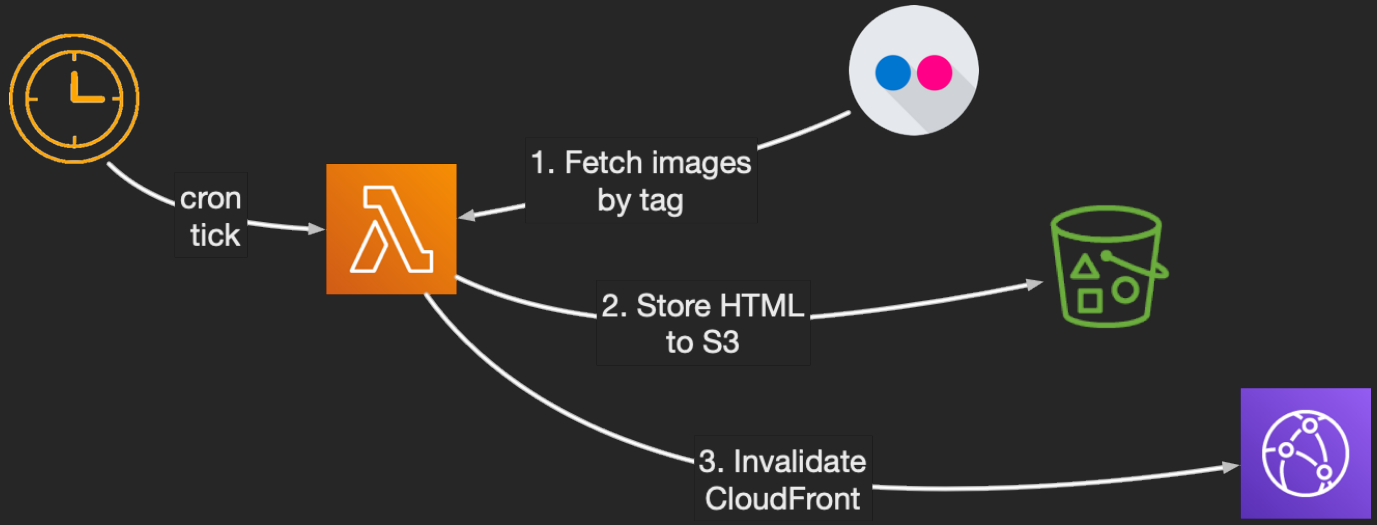
Project 365
My photo-a-day website

Project 365

Static website to display my photo-a-day picture for each day of the year.

- Hosted on S3
- CloudFront CDN
- Lambda/PHP function

Lambda/PHP function



Serverless configuration

```
functions:
  update:
    handler: src/actions/update.main
    layers:
      - {Ref: PhpLambdaLayer}
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    events:
      - schedule:
          name: project365-build
          rate: cron(0 */2 * * ? *)
```

Serverless configuration

```
functions:
  update:
    handler: src/actions/update.main
    layers:
      - {Ref: PhpLambdaLayer}
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    events:
      - schedule:
          name: project365-build
          rate: cron(0 */2 * * ? *)
```

Serverless configuration

```
functions:
  update:
    handler: src/actions/update.main
    layers:
      - {Ref: PhpLambdaLayer}
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    events:
      - schedule:
          name: project365-build
          rate: cron(0 */2 * * ? *)
```

Serverless configuration

```
functions:
  update:
    handler: src/actions/update.main
    layers:
      - {Ref: PhpLambdaLayer}
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    events:
      - schedule:
          name: project365-build
          rate: cron(0 */2 * * ? *)
```

Serverless configuration

```
functions:
  update:
    handler: src/actions/update.main
    layers:
      - {Ref: PhpLambdaLayer}
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    events:
      - schedule:
          name: project365-build
          rate: cron(0 */2 * * ? *)
```

main()

```php
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

main()

```
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

main()

```
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

main()

```
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

main()

```
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

Fetch photos from Flickr

```
$url = '?' . http_build_query([
    'api_key' => $this->flickrApiKey,
    'user_id' => $flickrUserId,
    'extras' => 'url_z, date_taken, owner_name',
    'method' => 'flickr.photos.search',
    'tags' => $year,
]);

$response = $this->client->get($url);
$data = json_decode($response->getBody(), true);
return $data['photos'];
```

Fetch photos from Flickr

```
$url = '?' . http_build_query([
    'api_key' => $this->flickrApiKey,
    'user_id' => $flickrUserId,
    'extras' => 'url_z, date_taken, owner_name',
    'method' => 'flickr.photos.search',
    'tags' => $year,
]);

$response = $this->client->get($url);
$data = json_decode($response->getBody(), true);
return $data['photos'];
```

Fetch photos from Flickr

```
$url = '?' . http_build_query([
    'api_key' => $this->flickrApiKey,
    'user_id' => $flickrUserId,
    'extras' => 'url_z, date_taken, owner_name',
    'method' => 'flickr.photos.search',
    'tags' => $year,
]);

$response = $this->client->get($url);
$data = json_decode($response->getBody(), true);
return $data['photos'];
```

Upload to S3

```
$s3 = new S3Client([
    'version' => 'latest',
    'region'  => getenv('AWS_DEFAULT_REGION')
]);

$s3->putObject([
    'Bucket' => $bucketName,
    'ACL' => 'public-read',
    'Key' => $filename,
    'Body' => $data,
    'ContentType' => 'text/html',
]);
```

Upload to S3

```
$s3 = new S3Client([
    'version' => 'latest',
    'region'  => getenv('AWS_DEFAULT_REGION')
]);

$s3->putObject([
    'Bucket' => $bucketName,
    'ACL' => 'public-read',
    'Key' => $filename,
    'Body' => $data,
    'ContentType' => 'text/html',
]);
```

Upload to S3

```
$s3 = new S3Client([
    'version' => 'latest',
    'region'  => getenv('AWS_DEFAULT_REGION')
]);

$s3->putObject([
    'Bucket' => $bucketName,
    'ACL' => 'public-read',
    'Key' => $filename,
    'Body' => $data,
    'ContentType' => 'text/html',
]);
```

Invalidate CloudFront

```
$cft = new CloudFrontClient([ .. ]);

$result = $cft->createInvalidation([
    'DistributionId' => $cloudFrontId,
    'InvalidationBatch' => [
        'CallerReference' => date('YmdHis'),
        'Paths' => [
            'Items' => ["$year.html"],
            'Quantity' => 1,
        ],
    ],
]);
```

Invalidate CloudFront

```
$cft = new CloudFrontClient([ .. ]);

$result = $cft->createInvalidation([
    'DistributionId' => $cloudFrontId,
    'InvalidationBatch' => [
        'CallerReference' => date('YmdHis'),
        'Paths' => [
            'Items' => ["$year.html"],
            'Quantity' => 1,
        ],
    ],
]);
```

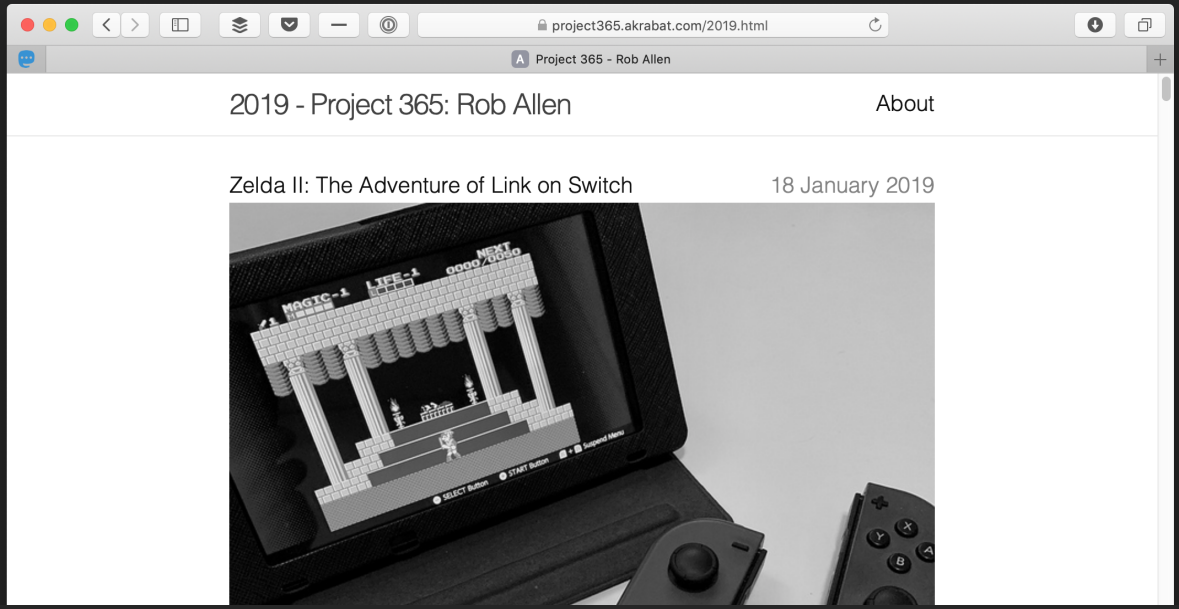
Invalidate CloudFront

```
$cft = new CloudFrontClient([ .. ]);

$result = $cft->createInvalidation([
    'DistributionId' => $cloudFrontId,
    'InvalidationBatch' => [
        'CallerReference' => date('YmdHis'),
        'Paths' => [
            'Items' => ["$year.html"],
            'Quantity' => 1,
        ],
    ],
]);
```

Invalidate CloudFront

```
$cft = new CloudFrontClient([ .. ]);

$result = $cft->createInvalidation([
    'DistributionId' => $cloudFrontId,
    'InvalidationBatch' => [
        'CallerReference' => date('YmdHis'),
        'Paths' => [
            'Items' => ["$year.html"],
            'Quantity' => 1,
        ],
    ],
]);
```

The finished website



To sum up

Resources

- https://akrabat.com
- https://www.martinfowler.com/articles/serverless.html

- https://github.com/akrabat/ow-php-todo-backend
- https://github.com/akrabat/project365-photos-website

- http://www.openwhisk.org
- https://aws.amazon.com/lambda/
- https://bref.sh

Thank you!